Heuristic analysis

Jinxiang Ruan

# Purpose

This project defines 3 problems in classical PDDL (Planning Domain Definition Language) in an Air Cargo transport system domain. After implemented the functions to represent the problem schema, the Planning Graph and the heuristic methods, this document is to analyse the execution performance between uninformed planning searches and heuristics searches.

# Uninformed non-heuristics searches

The uninformed non-heuristic planning was experimented with below search algorithms:

* Breadth First Search (BFS)
* Depth First Graph Search (DFS)
* Uniform Cost Search (UCS)

Metrics For uninformed non-heuristic Search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Problem | Search Function | Expansions | Goal Tests | New Nodes | Plan Length | Time (sec) | Optimality |
| P1 | BFS | 43 | 56 | 180 | 6 | 0.02764 | Yes |
| P1 | DFS | 12 | 13 | 48 | 12 | 0.00606 | No |
| P1 | UCS | 55 | 57 | 224 | 6 | 0.02982 | Yes |
| P2 | BFS | 3343 | 4609 | 30509 | 9 | 6.86852 | Yes |
| P2 | DFS | 582 | 583 | 5211 | 575 | 2.64048 | No |
| P2 | UCS | 4761 | 4763 | 43206 | 9 | 9.00185 | Yes |
| P3 | BFS | 14663 | 18098 | 129631 | 12 | 32.4266 | Yes |
| P3 | DFS | 627 | 628 | 5176 | 596 | 2.65447 | No |
| P3 | UCS | 17783 | 17785 | 155920 | 12 | 40.5475 | Yes |

# Execution time & Nodes Expansions

Sorting by execution time and nodes expansions from low to high: 1) DFS 2) BFS 3) UCS

That is to say, in terms of the time taken to reach goal state, **DFS** expands least number of nodes and is the fastest planning search among the three.

**Plan Length & Optimality**

**BFS** and **UCS** provide optimal solutions for all 3 problems while DFS doesn’t. And the plan length for DFS is much higher as compared to BFS and UCS.

# Heuristics searches

The heuristic planning was experimented using A\* search with below heuristics:

* h\_ignore\_preconditions (releax problem by ignoring preconditions)
* h\_pg\_level\_sum (planning graph sum of level costs)

Metrics For A\* heuristic Search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Problem | Heuristics | Expansions | Goal Tests | New Nodes | Plan Length | Time (sec) | Optimality |
| P1 | h\_ignore\_preconditions | 41 | 43 | 170 | 6 | 0.03387 | Yes |
| P1 | h\_pg\_level\_sum | 11 | 13 | 50 | 6 | 0.71442 | Yes |
| P2 | h\_ignore\_preconditions | 1450 | 1452 | 13303 | 9 | 3.63267 | Yes |
| P2 | h\_pg\_level\_sum | 86 | 88 | 841 | 9 | 64.3592 | Yes |
| P3 | h\_ignore\_preconditions | 5003 | 5005 | 44586 | 12 | 13.5388 | Yes |
| P3 | h\_pg\_level\_sum | 311 | 313 | 2863 | 12 | 326.130 | Yes |

# Execution time

The time taken to reach the goal state by h\_ignore\_preconditions is lower than h\_pg\_level\_sum, h\_pg\_level\_sum suffer from high computation hence it took longer to run.

**Nodes Expansions & Goal Tests & New Nodes**

h\_pg\_level\_sum outperforms h\_ignore\_preconditions by expansions, number of goal tests and number of new nodes.

**Optimality**

Both h\_ignore\_preconditions and h\_pg\_level\_sum provides optimal solutions for all 3 problems.

# Summary

Considering the balance of execution time reasonable number of nodes expanded, I recon A\* with h\_ignore\_preconditions heuristic is the best option, h\_pg\_levelsum do out performs in terms of number of nodes expanded, but suffer from high cost of computation and run much slower.

In terms of time taken, DFS is the fastest. However, it gives us a much higher plan length and the solution obtained by DFS is not optimal.

# Optimal Plans

**Problem 1**

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

**Problem 2**

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Load(C3, P3, ATL)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

**Problem 3**

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P1, ATL, JFK)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)